Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

A BRIEF INTRODUCTION TO COQ

The Coq Proof Assistant

February 17, 2020

Tanja Almeroth

Studienbereich DCSM
Hochschule **RheinMain**

TABLE OF CONTENTS

PROOF ASSISTANTS

THE COQ PROOF ASSISTANT

usage and applicability of the Coq-proof-assistant

- → developed since 1983
- → community in the industry and research
  https://coq.inria.fr/community
  https://github.com/coq-community
- → basic concepts of logic
- → functional programming

Figure: The Rooster. Source:[2]

OTHER PROOF ASSISTANTS

There are numerous proof assistants.

- → Isabelle
  https://isabelle.in.tum.de
- → HOL
  https://hol-theorem-prover.org

LOGIC

''As as matter of fact, logic has turned out to be significantly more effective in computer science then it has been in mathematics." [1]

LOGIC

Reliability in software is amplified by the costs of bugs by insecurity up to multiple levels.

- → basic tools from logic
- → precise claims about programs
- → functional programming methods of programming and logical reasoning about programs

FUNCTIONAL PROGRAMMING IN COQ

SYSTEM REQUIREMENTS

Coq runs on Linux, Mac OS and Windows.
There are a lot of development environments available.

→ **Proof General**
  an Emacs based IDE
→ **CoqIDE**
  is a simple stand-alone IDE
→ **coquille**
  a vim plug-in
→ and others

SYSTEM REQUIREMENTS

Coq runs on Linux, Mac OS and Windows.
There are a lot of development environments.

→ **Proof General**
  an Emacs based IDE
  generic interface for proof assistants developed by multiple universities

→ **CoqIDE**
  is a simple stand-alone IDE
  user-friendly, included in the official Coq-installation

→ **coquille**
  a vim plug-in
  an open-source project

PROOF GENERAL

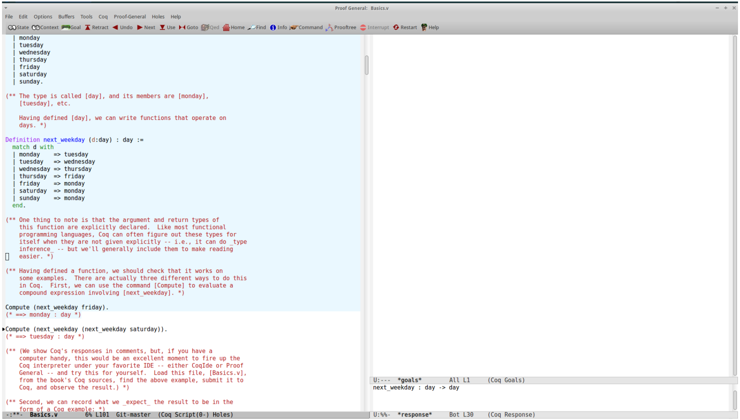


Figure: Coq interface in proof general.

COQIDE

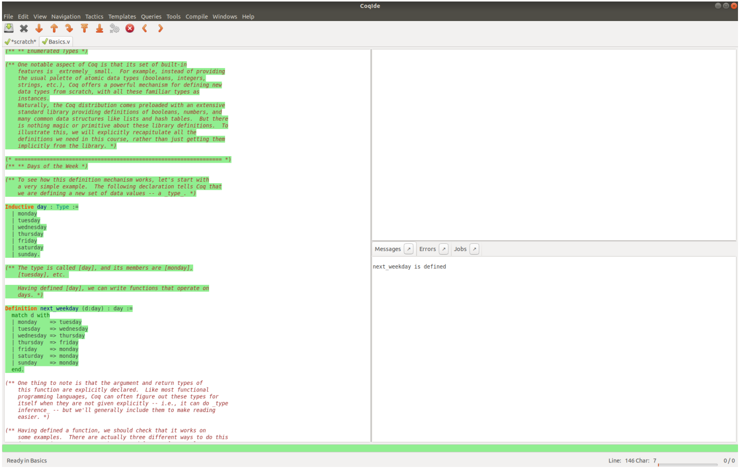


Figure: Coq interface in CoqIDE.

COQUILLE

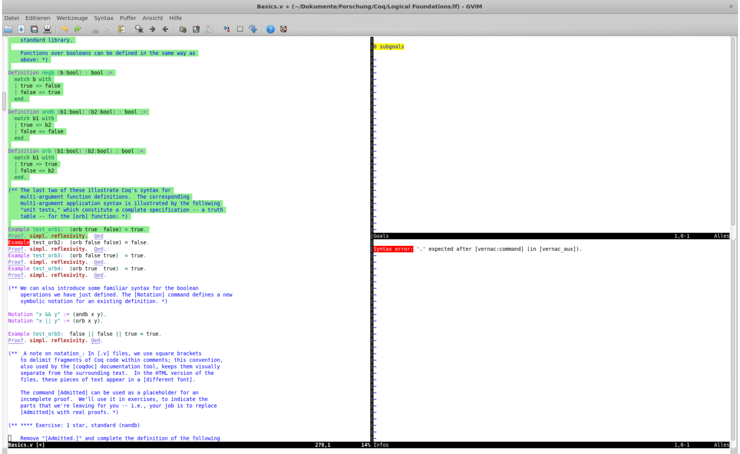


Figure: Coq interface in gvim using coquille.

FUNCTIONAL PROGRAMMING

Coq's functional programming language *Gallina's* sub-types

1. vernacular language (top-level interaction):
   e.g.: **`Theorem`**, **`Proof`**, **`Qed`**.
2. tactics language:
   e.g.: **`intros`**, **`exact`**
3. the *language of Coq-terms*:
   e.g.: **`for all A: Prop, A → A.`**

AN EXAMPLE

AN EXAMPLE

```
Inductive day : Type :=
  | monday
  | tuesday
  | wednesday
  | thursday
  | friday
  | saturday
  | sunday.
```

Listing 1: definition of a datatype called `day`

AN EXAMPLE

```
Definition next_weekday (d:day) : day :=
    match d with
    | monday    ⇒ tuesday
    | tuesday   ⇒ wednesday
    | wednesday ⇒ thursday
    | thursday  ⇒ friday
    | friday    ⇒ monday
    | saturday  ⇒ monday
    | sunday    ⇒ monday
    end.
```

Listing 2: declaration of a function on `day`

AN EXAMPLE

```
Compute (next_weekday friday).
```

Listing 3: call `Compute`

```
= monday: day
```

Listing 4: output

APPLICATIONS

APPLICATIONS

A platform for *modeling programming languages* and an environment for *formally certifying software and hardware*

- → PROSA a felxible open-source foundation for formally proven schedulability analysis. PROSA uses the Coq proof assistant and the SSREFLECT extension library [1]
- → Jasmin: High-Assurance and High-Speed Cryptography [1]

PROSA

PROSA - A COQ LIBRARY

Figure: RT-proofs-logo. Source: [3].

1. Formal Proofs for real-time systems (RT-Proofs)
2. project running between multiple research faculties (DFG project between INRIA, MPI-SWS, Onera, TU Braunschweig and Verimag, running from 2018 until 2020)
3. The PROSA library is where this development takes place.

PROSA - A COQ LIBRARY

to reproduce the PROSA - ECRTS'16 Artifact Evaluation by mechanically proofing the case study the following framework is provided online

utility	source
`Ubuntu 15.10, wily`	VM-Image
`Coq Version 8.5pl`	VM-Image
`Proof General 4.3pre131011`	VM-Image
`mathcomp-1.6`, `ssreflect`-extension	VM-Image
`PROSA v.01`	former version of the Coq spec and proof development of the RT-PROOFS project

PROSA - A COQ LIBRARY

to reproduce the PROSA - ECRTS'16 Artifact Evaluation by mechanically proofing the case study the following framework is provided online

my current working directory:
https://gitlab.cs.hs-rm.de/almeroth/prosa_working_dir
members: Steffen Reith
due to documentation including all listings

PROSA - FORMALLY PROVEN SCHEDUBILITY ANALYSIS

- »We conclude that the develop foundations are sufficiently felixble and powerful to support a large fraction of existing literature and real-time scheduling, without compromising readability.« [1]
- »Mechanized schedulability proofs are feasible, to the point that non-trivial multiprocessor schedulability analyses can be formalized in a reasonable time frame.«[1]
- **case studies**

PROSA - FORMALLY PROVEN SCHEDUBILITY ANALYSIS

a case study provided: Ciriniei's RTA for EDF [2, p.160, Figure 17.3]
definition and proofs of termination and correctness of Bertonga and Cicerei's RTA for EDF

Theorem (Main Theorem)

```
Theorem edf_analysis_yields_response_time_bounds :
     tsk R,
    (tsk, R) \\In edf_claimed_bounds ts →
    response_time_bounded_by tsk R.
```

Listing 5: `theorem_edf_analysis` [1]

PROSA - FORMALLY PROVEN SCHEDUBILITY ANALYSIS

practical specification: correctness of the proof

mechanized proofs available

- → correctness of the work-loaded-based interference bound for work-conserving schedulers
- → EDF-specific interference bound definition and proof of termination and correctness of Bertongna and Cicerei's RTA for FP scheduling
- → definition and proof of termination and correctness of Bertongna and Cicerei's RTA for EDF scheduling ($\approx 1'320$ LOC)

$\approx 13'070$ LOC

PROSA - FORMALLY PROVEN SCHEDUBILITY ANALYSIS

practical specification: correctness of the proof

mechanized proofs available

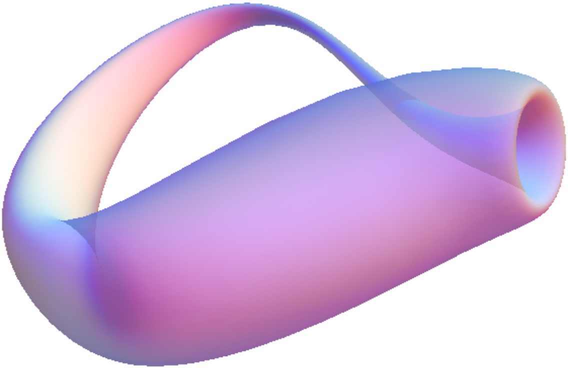
- → correctness of the work-loaded-based interference bound for work-conserving schedulers
- → EDF-specific interference bound definition and proof of termination and correctness of Bertongna and Cicerei’s RTA for FP scheduling
- → **definition and proof of termination and correctness of Bertongna and Cicerei's RTA for EDF scheduling** ($\approx 1'320$ LOC)

$\approx 13'070$ LOC

BREAK

BREAK - LET'S TALK ABOUT INSIDE AND OUTSIDE.

https://www.youtube.com/watch?v=SyD4p8_y8Kw



Klein Bottle with slight transparency, rendered with Mathematica 8 using the parameterisation provided by Robert Israel. Copyright by wikipedia, Creative Commons Attribution-Share Alike 3.0 Unported

PROSA - FORMALLY PROVEN SCHEDUBILITY ANALYSIS

Kasier's encapsulated EDF-scheduler [3]

- → real-time as in [1]
- → hierarchical (local and global distinction)
- → multi-core (in the real-time-sense)
- → EDF (earlines deadline first on a local level)
- → sporadic
- → disruptive

Main theorem: correctness of analysis although it is a deductive derivation

PROSA - FORMALLY PROVEN SCHEDUBILITY ANALYSIS

Definition

The *process* is defined as a sequential execution of a program on a processor. The execution ends after a finite number of steps. Therefore it corresponds to a finite execution of machine commands and is not separable.

Definition

A process is called *periodic* if it should be restated after a certain time called *the period*.
Otherwise a process is called *aperiodic* or *sporadic*. Furthermore, whenever as process is said to be *non-preemptive* the execution may not be interrupted between the beginning and ending of the process.
It is called *preemptive* if it may be interrupted after any instruction.

PROSA - FORMALLY PROVEN SCHEDUBILITY ANALYSIS

The **slotted prirority modell** from [1] with sporadic and periodic processes ([3]). Due to [1] the major requirement is said to be as in the following:

> *»If a system itself the execution of a real-time and non-real-time thread in alternate intervals the intervals in which real-time threads execute are scheduled to be in every l time unit, then it must be ensured that the interval begin at time t where $kl \leq t \leq kl + \epsilon \quad \forall \epsilon \geq 0$.«*

Moreover there is this requirement $\mathcal{B}$.

> *»For $L > \epsilon$ (for a suitable ϵ) for which the real-time thread schedule has asserted a real-time thread τ to be expected on the CPU, there must be a function of the method by which the minimum number of CPU cycles available to execute the instructions of τ can be determined.«*

PROSA-FORMALLY PROVEN SCHEDULABILITY ANALYSIS

Definition

$$\delta_p \in \mathbb{N} \quad \text{be a periodic disruptive process and} \tag{1}$$

$$\delta_s \in \mathbb{N} \quad \text{be an asporadic disruptive process.} \tag{2}$$

$$\text{Let} \quad P := \{1, \cdots, n\} \subset \mathbb{N}^n \quad \text{be an disruptable process.} \tag{3}$$

$$\text{Let} \quad \delta p_i, \quad \text{for} \quad i = 1, .., n \quad \text{denote the period and} \tag{4}$$

$$\delta e_i \quad \text{for} \quad i = 1, .., n \quad \text{denote the execution time.} \tag{5}$$

PROSA-FORMALLY PROVEN SCHEDULABILITY ANALYSIS

Moreover, we define the slotted priority model by Bollea with Kaiser's sporadic and periodic disruptive process with a discrete time in contrast to these authors. This chioce is due to the real-time model from [?] and the real-time kernel as in [1, chp. 5.3] and the exclusion model [1, p.12].

Definition

Time is $\mathbb{N}$.

```
...

Admitted.
```

OUTLOOK

- → lecture notes for Steffen Reith, incorporate review
- → incorporate linear temporal logic
- → Steffen Reith's review
- → incorporate Steffen Reith's review on the PROSA working directory
- → tutorial on `Gallina` and `SSreflect` (4 person days)
- → formally proof the Kaiser's EDF scheduler in PROSA hopefully less then ($\approx 1'320$ LOC)
- → see at what is the best way to approach the complete kernel

BIBLIOGRAPHY

- B. C. Pierce and A. A. de Amorim and C. Casinghino and M. Gaboardi and M. Greenberg and C. Hrițcu and V. Sjöberg and B. Yorgey
  »Software Foundations, Logical foundations, Volume 1«
  https://softwarefoundations.cis.upenn.edu/current/lf-current/index.html, 2019
- Coq- Project Website
  »The Coq Proof Assistant«
  https://coq.inria.fr , 2019-01-09
- RT-Proofs Website
  »Formal Proofs for Real-Time Systems«
  https://rt-proofs.inria.fr, 2020-14-02

BIBLIOGRAPHY

- Coq Integrated Development Environment - Offical Documentation
  »Coq Integrated Development Environment«
  https://coq.inria.fr/refman/practical-tools/coqide.html
- Proof General - Project Website
  »Proof General, a generic interface for proof assitants.«
  https://proofgeneral.github.io/, 2020-14-02
- Coquille - Andreas Werner's Fork
  »Coquille, a vim plugin.«
  https://github.com/Werner2005/coquille, 2020-14-02

BIBLIOGRAPHY: ONLINE

N. Giannarakis
Coq - Syntax Highlithing
»Personal GitHub Profile«
https://github.com/nickgian/thesis/lstcoq.sty, 2019-19-09

BIBLIOGRAPHY: REAL-TIME SYSTEMS

- R. Kaiser and K. Beckmann and R. Kröger
  »Echtzeitplanung«
  Handouts **https://www.cs.hs-rm.de/~kaiser/1919_ezv/6_Scheduling-handout.pdf** 2020-07-01

- J. W.S. Liu
  »Real-time Systems«
  Prentice-Hall, Inc., ISBN: 0-13-099651-3, 2000

- R. Kasier
  »Virtualisierung von Mehrprozessorsystemen mit Echtzeitanwendungen«
  PHD Thesis, Universität Koblenz-Landau, 11-02-2009

BIBLIOGRAPHY: REAL-TIME SYSTEMS

G. Bollella
»Slotted Priorities: Supporting Real-Time Computing Within General-Purpose Operating Systems«
PHD Thesis, Chapel Hill, 1997

M. Bertogna and M. Cirinei
»Response-Time Analysis for Globally Scheduled Symmetric Multiprocessor Platforms«
Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS 07)

BIBLIOGRAPHY: REAL-TIME SYSTEMS

F. Cerqueira and F. Stutz, and B. Brandenburg
»Prosa - ECRTS'16 Artifact Evaluation«
https://prosa.mpi-sws.org/releases/v0.1/artifact/,
2020-01-09

F. Cerqueira and F. Stutz, and B. Brandenburg
»PROSA: A Case for Readable Mechanized Schedulability Analysis«
https://www.mpi-sws.org/~bbb/papers/pdf/ecrts16f.pdf
Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS), 2016

BIBLIOGRAPHY: COMPUTER AND COMMUNICATION SECURITY

Almeida, José Bacelar and Barbosa, Manuel and Barthe, Gilles and Blot, Arthur and Grégoire, Benjamin and Laporte, Vincent and Oliveira, Tiago and Pacheco, Hugo and Schmidt, Benedikt and Strub, Pierre-Yves
»Jasmin: High-Assurance and High-Speed Cryptography«
Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2017